Slovenská technická univerzita v Bratislava, Fakulta informatiky a informačných technológií

Mobilné technológie a aplikácie

Cvičiaci: Ing. Miroslav Bahleda, PhD.

Cvičenia: Streda, 14:00

Kristián Rončkevič

1. Odkaz na verejný repozitár

https://github.com/kristianronckevic/MTAA_zad1.git

2. Použité knižnice

V našom zadaní sme sa rozhodli použiť knižnicu PySipFullProxy. Ide o voľne dostupnú Python knižnicu, ktorá umožňuje komunikáciu prostredníctvom SIP.

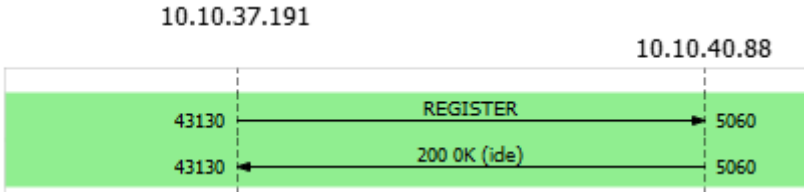
3. Použitý programovací jazyk

Naše zadanie sme sa rozhodli realizovať pomocou jazyka Python. Tento jazyk sme si vybrali z dôvodu jednoduchosti jeho použitia, ako aj jednoduchou implementáciou pre danú problematiku v spojení s knižnicou PySipFullProxy.

4. Implementácia vybraných funkcionalít

V našom zadaní sme implementovali všetky základné funkcionality.

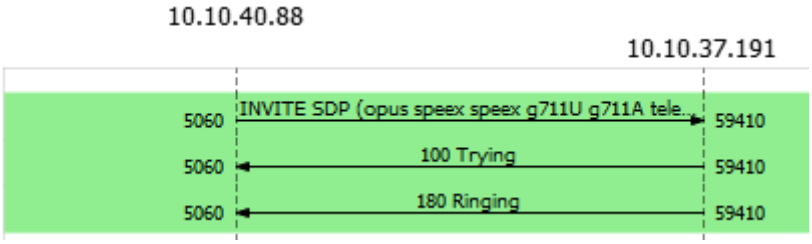
Registrácia:



Obrázok 1 Registrácia účastníka

Registrácia účastníka prebieha jednoducho. Registrovaný uzol zasiela požiadavku *REGISTER*, na ktorú mu server odpovedá *200 OK.*

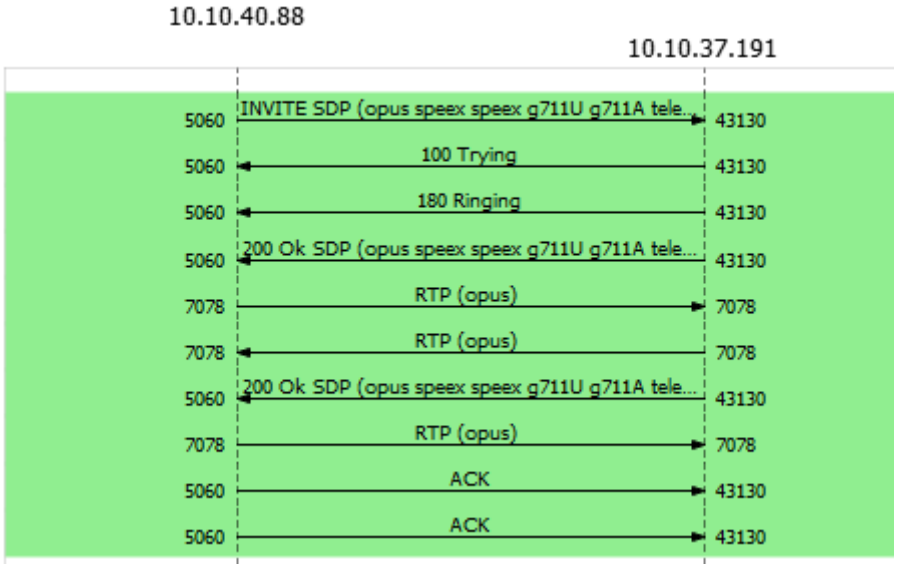
Vytočenie a zvonenie:



Obrázok 2 Vytáčanie uzla

Na tomto obrázku môžeme vidieť priebeh vytáčania a zvonenia pri pokuse o hovor medzi 2 uzlami. Ako prvá sa zasiela požiadavka *INVITE*. Táto požiadavka hovorí o tom, že uzol 10.10.40.88 sa chce telefonicky spojiť s uzlom 10.10.37.191. Odpoveďou je *100 Trying*, ktorý je priamou odpoveďou na *INVITE,* a teda že uzol *.191 obdržal *INVITE* a je pripravený na hovor. Odpoveď *180 Ringing* znamená, že danému telefónu začína zvoniť telefón (prebieha vytáčanie).

Prijatie hovoru a funkčný hlasový hovor:

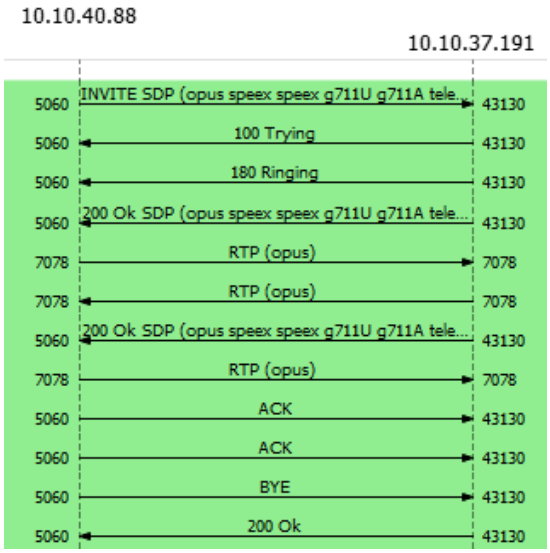


Obrázok 3 Funkčný hlasový hovor

Po zodvihnutí hovoru je zaslaná cieľovým uzlom odpoveď *200 OK.* Ide o potvrdenie, že uzol prijal hovor. Následne sú medzi uzlami vymieňané hlasové dáta, ktoré sú označené ako *RTP (opus).*

Ukončenie hovoru (prijatého/neprijatého):

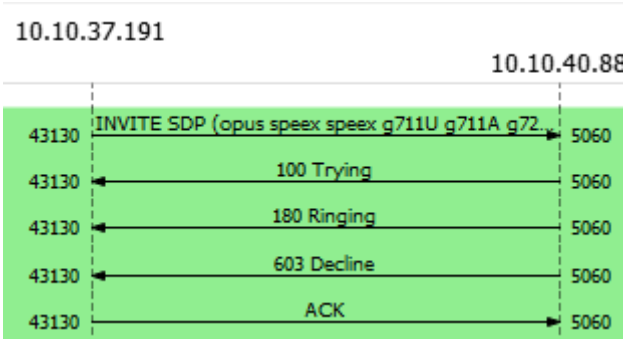
Ukončenie hovoru môže nastať dvoma spôsobmi.



Obrázok 4 Ukončenie prijatého hovoru

Ukončenie prijatého signalizuje požiadavka *BYE.* Tá je zaslaná uzlom, ktorý inicializuje koniec hovoru. Ako odpoveď je znova zaslaná *200 OK,* ktorá symbolizuje potvrdenie ukončenia hovoru.

Druhým spôsobom je ukončenie neprijatého hovoru, a teda ukončenie hovoru v procese vytáčania.

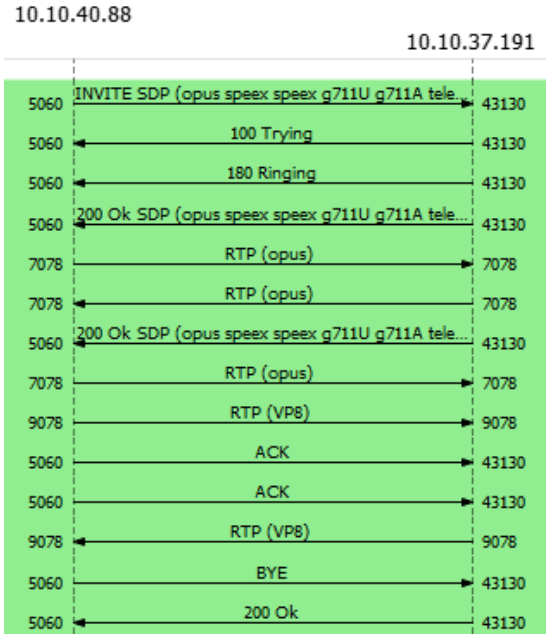


Obrázok 5 Hovor ukončený počas vytáčania

Vtedy zašle volaný uzol odpoveď *603 Decline*, a teda odpoveď, že hovor sa rozhodla neprijať. Volajúci uzol odpovedá *ACK*.

Po implementovaní povinných funkcionalít sme sa rozhodli implementovať aj doplnkové funkcionality.

Realizácia videohovoru:



Obrázok 6 Priebeh video hovoru

Priebeh video hovoru je z veľkej časti identický. Hlavnou zmenou je, že uzly si vymieňajú 2 typy dát: hlasové dáta, ktoré sú v tomto flow-e reprezentované ako *RTP(opus)* a video dáta, ktoré sú v tomto flow-e reprezentované ako *RTP(VP8).*

Denník hovorov:

Ako ďalšiu bonusovú funkcionalitu sme sa rozhodli implementovať denník hovorov. Ten sleduje zasielané požiadavky, ktoré sme si rozobrali už skôr v práci, a na ich základe teda vieme povedať, kedy je hovor inicializovaný, kedy je prijatý a kedy ukončený.

```
*********************************
*******Inicializujem hovor*******
23/02/2022 21:11:59
 Volaný: kristian@10.10.40.88
 Volajúci: kristian_telefon@10.10.40.88

23/02/2022 21:12:01
**********Hovor prijatý**********

23/02/2022 21:12:05

**********Hovor ukončený*********
*********************************
```

Obrázok 7 Formát 1 volania v denníku hovorov

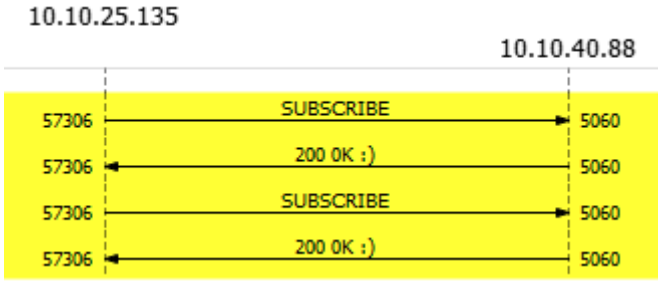
Na obrázku vyššie môžeme vidieť zvolený formát výpisu hovoru.

Každý výpis začína informáciou o tom, že hovor je inicializovaný. Táto informácia obsahuje taktiež dátum a čas inicializácie hovoru, názov volajúcej, ako aj volanej strany.

Druhou časťou výpisu je informácia o dátume a čase prijatia hovoru. Ako posledná je časť s dátum a časom ukončenia hovoru.

Konferenčný hovor:

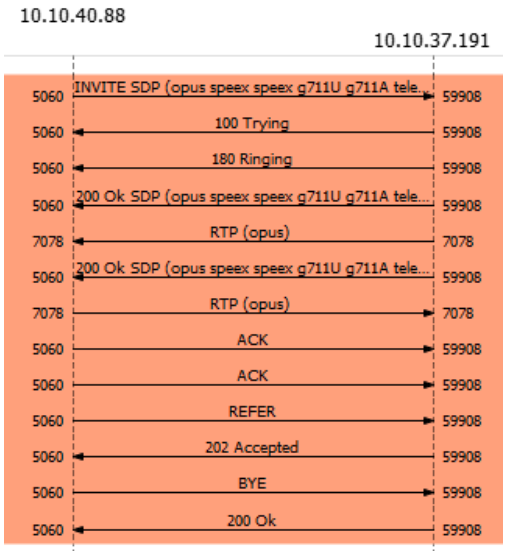
Ďalšou bonusovou funkcionalitou je konferenčný hovor. Ten sme vykonali medzi 3 zariadeniami. Výmena informácií medzi zariadeniami prebieha ako pri štandardnom hovore. Výnimkou je výmena požiadaviek *Subscribe* medzi jednotlivými účastníkmi.



Obrázok 8 Výmena požiadaviek Subscribe medzi účastníkmi konferenčného hovoru

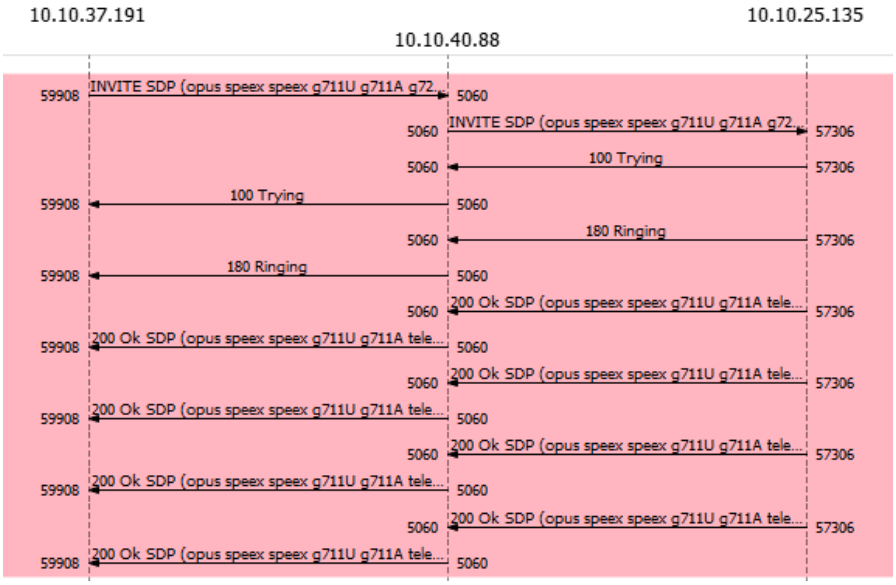
Presmerovanie hovoru:

Ďalšou bonusovou úlohou bolo presmerovanie hovoru. Pri presmerovaní nastávajú 2 zmeny. Prvou je požiadavka *Refer*, ktorá oznamuje presmerovanie na iného účastníka.



Obrázok 9 Ukážka požiadavky Refer na presmerovanie na nového účastníka

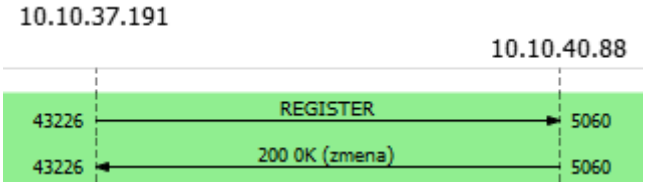
Druhou je samotný hovor medzi presmerovanými uzlami. Ten neprebieha priamo medzi 2 účastníkmi, ale prostredníctvom pôvodného uzla (3. strany).



Obrázok 10 Priebeh hovoru medzi presmerovanými uzlami

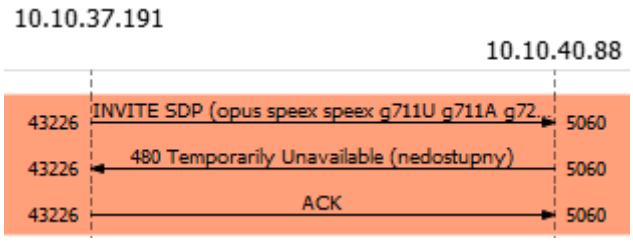
Úprava stavových kódov:

Ako poslednú bonusovú úlohu sme si vybrali úpravu stavových kódov, ktoré sú zasielané medzi uzlami.



Obrázok 11 Zmena odpovede 200 OK pre registráciu

Ako prvú sme zmenili odpoveď *200 OK* na požiadavku *Register.* Aby sme zachovali konvencie, zmenili sme ju na *200 OK (zmena).*



Obrázok 12 Zmena odpovede Temporarily Unavailable

Ako druhú sme vykonali zmenu kódu *480 Temporarily Unavailable.* Rovnako ako v prvom prípade, sme chceli zachovať konvencie a stavový kód ma nový formát *480 Temporarily Unavailable (nedostupny).*

5. Použitá ústredňa

Zadanie sme sa rozhodli testovať pomocou ústredne Linphone. Tá nám vo svojej free verzii poskytuje všetky nástroje, ktoré boli potrebné pri realizovaní povinných aj bonusových funkcionalít.

6. Kompilácia programu a jeho fungovanie

Pre tvorbu programu sme použili IDE PyCharm. V ňom sme vytvorili funkciu, ktorá si zo systému vyberie hostname a funkčnú IP adresu, ktorú bude následne používať.

```
hostname = socket.gethostname()
logging.info(hostname)
ipaddress = socket.gethostbyname(hostname)
```

Obrázok 13 Zistenie hostname-u a IP adresy

Následne vytvárame server pomocou funkcie *UDPHandler* z nami implementovanej knižnice.

```
server = socketserver.UDPServer((HOST, PORT), sipfullproxy.UDPHandler)
server.serve_forever()
```

Obrázok 14 Vytvorenie servera

Tá nám nakonfiguruje a vytvorí náš server. Funkcia *serve_forever()* zaručí, že server bude „bežať“ do doby, kým neukončíme program.

Obsluha programu je teda úplne jednoduchá a stačí nám ho len skompilovať.